

Aus 920010160451

1

A Method for Constructing and Caching a Chain of File Identifiers and Enabling Inheritance of Resource Properties in File Systems

Field of the Invention

5

The present invention relates generally to a technique for controlling access to file system resources using externally stored attributes. More specifically, this invention describes a technique in which the externally stored attribute, such as an authorization security policy, uses an array of file system identifiers to determine access to a file system resource list in this array.

10

Background of the Invention

Many of the UNIX operating systems of today support the generation of file identifiers for file system resources. A FID is a binary representation that uniquely defines a physical file system object that resides in a file system. The FID is typically a stream of bytes of arbitrary length that is commonly as small as eight to ten bytes in size. The contents of the FID bytes are often numerical in nature with the first set of bytes holding an index or "inode" number and the remaining bytes holding a generation or use instance of the inode.

20

Fids are used in present day operating systems for the implementation of network file sharing services. A file server process running on a file system server machine housing data will obtain a FID for a file when a client machine on a network searches for the file by name. The server will then return the FID to the client. The client sends the FID in subsequent requests to the server to perform operations on the file such as reading or writing data. The server uses the FID to quickly find the file system object's data structure and perform the operation. Thus in a network file system implementation, the FID acts as a alternate representation that can be quickly mapped to the object's defining data structure which often is a inode/vnode. Inode and vnode are used synonymously in this description. The vnode is an in-memory object that usually has a short lifetime and is frequently recycled for use by other accessed file objects. However the FID allows fast construction of a vnode for the unique physical file system object it describes.

30

The numerical nature, typically small size, and unique mapping to an individual file system object make the FID a powerful association tool. Given that the FID represents a single instantiation of an object, it also represents a unique mapping for any of the potential pathnames or alternate descriptors that can be used by an application to access the object. Thus a FID can be used as an efficient bi-directional mapping equivalent of file system object and any of its names.

FIDS are a powerful concept as their finite size and numeric nature make them suitable for efficient storage and searching. Once a FID is obtained through a file resource name to FID translation, it can be used to quickly obtain access to the actual underlying file system object data structures and access methods. The properties of FIDs make them useful in distributed file system server implementations for projecting a "file handle" out to client systems. The client can then provide the handle back to the server so it can quickly locate the associated resource's underlying data structure and perform operations requested by the client.

As has been mentioned, FIDS are beneficial in UNIX distributed file system implementations. Servers generate and provide FIDS to clients for use in subsequent transactions against the file resource corresponding to the FID. For example, writing data to the file. In the above example of /usr/local/bin/date, the client would first lookup the file by its name (/usr/local/bin/date) at the server. The server would return a FID for date along with other resource attributes for date. The client would then contact the server for additional operations providing the FID to the server. The FID allows the server to locate and operate on the underlying file resource much faster than if it had to process the file path name on each call from the client. If after accessing date, the client desired to create a new file called time in /usr/local/bin, it would first contact the server with the name /usr/local/bin to get the FID for bin. Next it would send another request to the server to create time in the directory bin using bin's FID. Situations like this where components higher up in a directory tree end up being accessed are actually common. One optimization in the above procedure would have been for the server to return the FIDS for all the components in the pathname leading to date as well as vi when the client first accessed /usr/local/bin/date by name at the server. This would have saved the server call

to get the FID for /usr/local/bin. In a networked environment the savings could be significant. If there are other clients performing similar accesses to the server then even more calls are saved. Thus in the above example, the efficiency of the network file system protocol could potentially be improved by having the server return an array of
5 FIDS for all the components along the file path provided in an initial client lookup request.

FIDS have other potential uses outside of network file system implementations as well. File resources in UNIX are represented in an inverted tree name space with the / character separating the components along the file path to the file resource. Each
10 component represents a file system resource. The terminating component represents the target file resource of the path and the prior components represent a series of directories leading to the resource. For example, a path of /usr/local/bin/date specifies the path to the file resource named date which resides in the directory /usr/local/bin. The path contains four components with at least three of them being directories and the fourth terminating
15 component (date) representing several possible file resource types. Resource types include directories, files, links, and special device files. The FID relationship exists at the individual file resource level. Thus in the example of /usr/local/bin/date, there are four associated FIDS.

FIDS have a unique one to one relationship to the physical file resources in a file
20 system. In addition, the UNIX operating system can produce FIDs for the many forms of named or non-named methods an application program might use to access them. This property makes FIDS a good association tool for applying external attributes to file resources. One of the resource's well known names such as its full pathname starting at the file system root are used in a database of attributes which might be anything from
25 extended information about the resource to rules describing additional security protections on the resource. Such a method is described in IBM Patent Application AUS9-2000-0672 titled "A Method for Attachment and Recognition of External Authorization Policy on UNIX File System Resources". In the described methods security policy is placed on file resources using the full path name starting at the root of
30 the file system. For example, /usr/local/bin/date. This name known as a protected object name (PON) is used as the name for locating the defined attributes. The PONs are then

processed into their associated FIDs thus creating a mapping between the resource's FID and the defined external attributes. When file resources are accessed by applications, the file specification used in the access is converted into a FID, which can then be used to find the attributes in the external database.

- 5 An extension of those methods would be to traverse up the directory path of the resource towards the root of the file tree searching for components having attached attributes when the target of the access has no attached attributes. If an external attribute is found in the search, then it could be inherited and be treated as the effective attribute for the accessed file system resource. For both of the above examples, there exists an
- 10 opportunity for operational improvement if an array of FIDS representing all the components along a file path to a file resource are available. To utilize this feature, methods that produce FIDS require techniques to create such an array of FIDS. The cost of producing an individual FID requires a lookup operation on the file system resource name. This lookup operation requires a relative degree of computational expense.
- 15 Therefore techniques to produce such an array need to be efficient to minimize the performance impacts to the system. With efficient techniques and methods to produce a FID array, systems, which use FIDs in their processing, can exploit new mechanisms for more efficient operation or to support new features.

Summary of the Invention

It is an objective of the present invention to provide a method for producing an array of FIDs representing the full file path name starting the root of the file system for a file resource.

It is a second objective of the present invention to provide a method for storing a generated FID chain.

It is another objective of the invention to provide for retrieval of stored FID chain data on subsequent file system accesses involving the same directory path represented by a cached FID chain.

It is another objective of the present invention to provide a method for invalidating a cached FID chain entry when operations occur that might render stored entries stale.

It is another objective of the present invention to enable the capability of implementing inheritance along a file resource's directory path to the resource.

It is another objective of the invention to provide for the association, recognition, and processing of external attributes utilizing an array of system object file identifiers.

This invention describes a method for constructing an array of file identifiers "FIDs" also known as a FID chain, which represents all the directory components of a file system pathname to a file system resource. These components represent the individual file system resources in the form of directories, which must be traversed along the file system hierarchy in order to access the file system resource represented by the terminating component in the path. A FID for a component has the notation FID(component). For example, a file path of /usr/local/bin/date would result in FID chain with FIDs such as FID(usr), FID(local), FID(bin). FID(bin) would be the terminating FID of the directory path. "date" would be the terminating component of the entire path specification including date. Once a FID chain is constructed, it is inserted into a storage location known as a "FID chain cache". The terminating FID of the chain acts as the search key to locate a FID in the cache. In the above example of /usr/local/bin/date, FID(bin) would be the search key.

Another feature of the present invention is a method to keep the cached information from becoming stale. In this process, mechanisms are provided to invalidate cached FID chains in the event an operation occurs which could affect the file system name space. For example if /usr/local was renamed to /usr/loc, then the name space would change making a cached FID chain for /usr/local/bin stale. To enable the use of FID chains, this invention provides techniques for retrieving FID chains and for processing the chains to implement inheritance. With this inheritance, a component of the FID chain would inherit the security policy that governs another component of the FID chain.

The invention is implemented in the context of a resource security classification system. This system will also be referred to as an SCS. In the example system, a database of security classifications is defined for named system resources. The names used for the resource are full file system path names of those resources, starting at the root of the file name space. Examples of full path names would include /usr/local/bin and /home/joe/projectX/datafiles/dfile. These names will also be referred to as defined names or DNs in subsequent text. The security classification database (SCDB) contains security categories for the DNs. Example security categories might include, public, confidential, internal use only, and top secret. A security classification category will further be referred to as an SCC.

The techniques of the present invention involve constructing and caching a chain of FIDs that represent the directory path to a system resource. First the target resource name is processed using a sequence of operating system services. Typically a lookupname() service is provided which given a valid object path name returns a handle to the object and a handle to the object's owning directory. In a UNIX system, this handle is commonly called a vnode. The owning directory handle can then be used to obtain a file identifier (FID) for the owning directory. This FID constitutes the first FID in the chain and will also act as the lookup key in the cache. The process next finds the directory's parent. This parent is found using a vn_lookup() method, which takes a directory vnode and a well-known name designation for directory's parent directory. It returns a handle (vnode) to the directory's parent. An example call might be:

```
Code = VOP_LOOKUP(handle, "..",&parentHandle);
```

With this obtained parent handle, a FID is obtained using a VOP_FID() service. This FID is added to the chain. The process repeats until the root of the system's file tree is reached. This result is a chain or array of FIDs representing the full path name of the directory containing the accessed object. The chain begins with the directory FID closest to the accessed object and ends with the FID representing the root of the file tree. Once constructed, the FID chain, which forms a binary representation of a directory path name, is placed in a cache. The cached chain remains valid unless a rename operation occurs on any FID components in the chain. On an object access, a FID for the object's owning directory is obtained. This FID, which represents the first FID in a FID chain, is then used to search the FID chain cache. If a match is found, then the cached FID represents the path name for the access object. With this found chain, the expense of constructing the additional component FIDs for the object's owning path is avoided.

The functional goal of the example embodiment is to monitor accesses to file system resources and audit the use of the resources based on the classification level. The system seeks to utilize the feature of inheritance when the accessed target resource does not have an externally associated classification. For example, if the path /usr/local was classified as internal use only, then an access to /usr/local/bin/date would be considered an access to internal use only data assuming /usr/local/bin and /usr/local/bin/date had no defined classification. Finally, the SCS reports statistics on the count of accesses to each classification category. In an actual implementation, such statistics might be maintained in memory for retrieval through application programming interfaces (APIs) or the data may be logged on non-volatile media such as a disk.

In summary, the functions of the example embodiment are to allow the definition of security classifications on file system resources, monitoring accesses to the resources, and recording statistics for the accesses based on the classification category, which applied to the access. The SCS uses the techniques described in the present invention to achieve its functional goals. In addition, the SCS example implementation is based on the use of FIDs as an association object to the DNs and SCC information defined in the SCDB. FIDs are also used in the monitoring component of the SCS to recognize the existence of defined security classification data on a file system resources being accessed or on one of the file system resources associated with the directory components traversed

along the file system path name to the accessed resource. The techniques of this method are described in IBM patent application AUS9-2000-0672.

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225

Description of the Drawings

Figure 1 shows a flow diagram of the steps in the example SCS embodiment to process a DN from the SCDB into a FID to DN mapping database.

Figure 2 shows a flow diagram of the steps in the example SCS embodiment for
5 monitoring file system resource accesses, finding effective security classifications for the resource, collecting statistics, and potentially invalidating cached FID chain information.

Figure 3 is a flow diagram of the steps in the example embodiment to check for the presence of attached security classifications on an accessed file system resource. In the absence-attached properties, a search along the resource's associated FID chain in
10 order to discover the presence of attached properties, which should be inherited.

Figure 4 is a flow diagram of the steps to obtain a FID for an accessed resource along with a FID chain for the directory path where the accessed resource resides provided a valid file name specification for a file system resource.

Figure 5 is a flow diagram of the steps to find a FID chain in a FID chain cache
15 (FCC) provided a FID specification of the terminating component of a directory path specification.

Figure 6 is a flow diagram of the steps to invalidate one or more entries in the FID chain cache.

Figure 7 is a diagram of an example high-level architecture of the described SCS
20 embodiment.

Figure 8 depicts a pictorial representation of data processing system, which may be used in implementation of the present invention.

Detailed Description of the Invention

The present invention is described in the context of a UNIX operating system. The described steps would also be applicable to UNIX like operating systems such as Linux and could be generally applied to systems, which provide internal FID like concepts to describe file system resources. The invention describes a set of techniques and logic flows for constructing arrays of FIDs also known as FID chains. It further describes techniques to cache constructed FID chains for subsequent reuse to avoid the costly processing involved in constructing a FID chain from scratch. In addition, a technique is provided using FID chains to enable the feature of inheritance of properties from file system resources along the directory path to the destination resource. These techniques are described in detail within the above example described Security Classification System (SCS). The example system employs FIDs as a fundamental association tool between actual file system resources and fully defined names known as DNs, which are used to name the records in the Security Classification Database (SCDB). The techniques for FID association and the use of FIDs at resource access time for recognition of external named properties can be found in IBM Patent Application AUS9-2000-0672.

Referring to the implementation of the invention in Figure 1, described is the set of steps involving the processing of SCS records which consists of DNs and their defined security classification category (SCC). The flow of steps shows the processing of each DN/SCC pair into a mapping database, which hold FID to DN mappings. Each mapping also contains the SCC attribute. The flow starts in step **100** by providing the DN and SCC as inputs. In step **101**, a FID is obtained for the DN. The FID is generated using programming interfaces (also called services) resident in the native operating system. FID generation techniques are described in detail in a co-pending patent application docket no. AUS9-2000-0672 and incorporated herein by reference. The technique then proceeds to step **102** where a FID to DN mapping containing SCC data is added to the FID to DN mapping database. This database could reside in the memory of a computing or process system or might reside on non-volatile storage such as a disk. The flow then ends with step 103.

Figure 2 presents the processing steps that occur when an application attempts to access a file and the SCS monitoring component intervenes in the access. The process begins with step **200** where the SCS invokes this process and provides as input the path name used to access the underlying file system resource and the type of operation on the resource. The pathname would be one of the many possible variations supported by UNIX such as a full name (e.g. /usr/local/bin/date) or a relative name (../bin/date). The operation describes what type of action is being performed on the resource. Examples of operations could include read, write, change permissions, or rename. The flow proceeds to step **201** where the FID for the target of the access (date) is obtained along with a FID chain for the directory (/usr/local/bin), which is traversed to get at date. The processing of this step is further detailed in Figure 4 of the present invention. Step **202** searches the FID to DN/SCC mapping database to locate an SCC definition. Step **202** finds the effective SCC and DN from the FID or FID chain. The search of the mapping database is described in greater detail in Figure 3 of the present invention. If no SCC is found in step **203**, the process proceeds to step **204** where the SCC definition is defaulted to the category "unclassified" and the DN is set to the path used in the file system resource access. Step **204** would then proceed to back to the main processing path indicated by step **205**. If in step **203** a SCC was found and returned in the mapping database search, the flow proceeds directly to step **205** where the accounting data maintained by the SCS is updated. An update could include incremented the count of accesses for the given classification and perhaps adding a record to the monitoring log consisting of the DN corresponding to the access and the classification defined for the DN. The process then proceeds to step **206** where the desired operation is checked to determine if the operation could affect the file system name space. An operation such as a rename on a directory could have such an effect and potentially render a cached FID chain in the FID chain cache (FCC) as stale. For example, if /usr/local where renamed to /usr/loc, then any FCC entry containing a FID for local would be stale and would require invalidation. The invalidation logic is detail in Figure 6. If in step **206**, the operation requires an FCC flush, step **207** is invoked to flush the entries related to path. Step **207** then proceed back to the main logic flow which ends with step **208**. If in step **206**, the operation does not affect the

file system name space, then logic proceeds directly to stop **208** where the process of Figure 2 ends.

Figure 3 describes the flow of steps to search for an entry in the FID to DN/SCC mapping database provided a FID (tgtFid) representing the a file system resource target which in this example is the final target of a file system resource access. In the previously presented example of /usr/local/bin/date, the target would be date. In addition, the FID chain for the directory path is the target described by tgtFID. In the example of /usr/local/bin/date, the FID chain would be the array of FIDs for /usr/local/bin/date. The FID chain is actually organized in reverse order. In other words, using the notation of FID(component name) to represent a FID for a corresponding directory path name component; the FID chain for /usr/local/bin would be represented as FID(bin), FID(local), FID(usr). This order facilitates a search for an SCC association starting with the directory closest to the accessed target (tgtFid) and proceeding up the directory path toward the root of the file system name space. The effect is that if an SCC mapping is found for a component FID along the directory path, it is the SCC mapping "closest" to the accessed resource (tgtFid) which acts as the effective inherited SCC for tgtFid. The set of steps in the search begins by providing inputs in step **300**. The process moves to step **301** where the mapping database is searched for a mapping entry corresponding the FID (tgtFID), which in the example path would be date. The details of the search mechanisms could be implemented in a variety of ways and are not important in describing the techniques of the present invention. The flow proceeds to step **304** where there is an inquiry whether an SCC was found for tgtFid. If so, the processing ends in step **305** with the found SCC and DN being returned to invoker of the steps of Figure 3. If in box **304**, an SCC is not found, then the processing proceeds to step **306** where the first FID in the FID chain (FID(bin)) is taken from the chain. The logic then proceeds to step **307** where the mapping database is again searched looking for a SCC/DN mapping using the FID from the chain. If in step **308**, an SCC and DN are found, the process proceeds to step **309** where the "inherited" SCC/DN information is returned to the invoker. If an SCC is not found, then the process proceeds to step **310** which checks to see if there are more FIDs in the FID chain. If not, the processing ends with step **311**, which returns to the invoker with no SCC being found. If in step **310**, more FID chain

FIDs exist, then the flow proceeds to step **312**, which retrieves the next FID in the chain. Processing then returns to step **307** where the mapping database is searched using the new FID in the chain.

Figure 4 shows the flow of steps to create and return a FID for the file system resource described by the provided path. In addition, this step produces and returns a FID chain representing the full directory path starting at the root of the file system name space, which leads to the resource of path. The flow starts with step **400** by providing the path name of a file resource (path). The process moves to step **401** where the UNIX operating system function `lookupname()` is used to obtain an underlying data structure, which is called a “vnode”. The service “`lookupname()`” takes a path name as input and returns the vnode and optionally can also return a vnode for the parent directory where the file system resource described by path resides. In the presented logic flow, `lookupname()` is used to obtain a vnode for both the resource described by path (the target) as well as the parent directory (the parent) where the target resides. The flow continues to step **402** where the vnodes from step **401** are used to obtain the FID (`tgtFid`) for the target and the FID (`dirFid`) for the parent. The FID is obtained using the operating system function `VOP_FID()` that takes the vnode as an input and returns a FID. `VOP_FID()` is a function found on most every commercial UNIX implementation today. If a `VOP_FID()` function does not exist, then alternative methods can be used to construct a FID such as those described in a co-pending patent application docket number AUS9-2000-0672. With the obtained FIDs, the processing proceeds to step **403** where the FID chain cache (also known as the FCC) is consulted to determine if a chain corresponding to the directory path ending with the directory component represented by `dirFid` is already cached. The steps of searching the FCC are detailed in Figure 5 of this description.

The FCC is organized such that the terminating directory component of a chain of FIDs is used as the search key. For the example, file path name of `/usr/local/bin/date`, “bin” is the terminating directory component of the path leading to the file resource “date”. Correspondingly, the FID for “bin” (`FID(bin)`) is the key in the cached chain for `/usr/local/bin`. Given that a FID is a unique representation of a file system resource, it uniquely represents the directory component bin, and until the file system name space is

changed, it also uniquely represents the location of bin in the file system name space (/usr/local). As a result, finding an entry in the FCC for the key of FID(bin), represents a hit on the FID chain that represents the full path of directories including "bin", starting from the root of the file system name space (root). In the example of, /usr/local/bin/date, the associated FID chain for bin, represents the complete chain of directories starting at the root and leading to the file system resource date. In other words it is the FID chain for /usr/local/bin.

After consulting the FCC, the process moves to step **404** where there is a determination whether a FID chain was found in the FCC. If a FID chain was found in the FCC, then the FID for the target resource and the FID chain for the parent have been obtained, and the flow proceeds to step **416** where the FID and FID chain are returned to the caller. If the test in step **404** results in a FID chain not being found in the FCC, then the flow proceeds to step **405** which begins the steps to construct a FID chain for the parent directory (the parent) where the target specified by path (the target) resides. The construction of a FID chain for the parent begins in step **405** by starting the FID chain with the FID of the parent (dirFid). In the example path of /usr/local/bin/date, dirFid is the FID for bin represented by the notation FID(bin). As was previously described, FID(bin) will represent the key for this FID chain and will also represent the first FID entry in the FID chain. The logic flow now proceeds to step **406** where the temporary vnode variable "tVnode" is introduced to facilitate the described logic flow. The tVnode is set equal to the vnode for the parent represented as dirVnode. The process now moves to step **407** where tVnode is tested to see if it is the root directory of the file system name space.

The root directory is the top most directory in the UNIX file system name space and is represented by the notation /. In the example of /usr/local/bin/date, the first / is the root and the directory usr resides in the root directory. Those skilled in the art would recognize that testing a vnode to see if it is a root directory involves a test in the UNIX kernel of the vnode against the global kernel symbol for the root vnode, which is available to UNIX kernel applications and device drivers. This can also be determined by obtaining the vnode for the path "/", and comparing it against the vnode, tVnode. If the result of step 407 is that tVnode is the root directory, then all the FID components of the

FID chain have been obtained and added to the chain. The construction of the chain has been completed and the logic moves to step **408**, which proceeds to step **415**. In the presented flow of steps for the present invention, the FID for the root directory is not physically added to the FID chain. The flow could be extended to include that step.

5 However, that is not required since reaching the end of the entries in a FID chain implies that the next entry would be the FID for root (FID(/)). Thus root FID can be omitted from the chain saving the space to store it.

If the test in **407** does not find tVnode to be the root directory as would be the case in the example path of /usr/local/bin/date where tVnode is currently the vnode for bin, then the logic proceeds to step **409**. This step **409** obtains the vnode designated as PV, which is the directory of the parent. In the presented example path, /usr/local/bin/date, step **409** is currently seeking to obtain the vnode for local. The vnode for the parent directory of the currently set directory represented by tVnode is obtained using the UNIX VOP_LOOKUP() operation, which takes as inputs, the vnode of the

10 directory to look in, and the string for the name of the file system resource to search for. It returns the vnode for the name on which it is searching. The UNIX string notation which represents the parent or previous directory up the file system name space is "..". Therefore, to find the parent directory for tVnode, VOP_LOOKUP() is invoked as follows;

20 VOP_LOOKUP(tVnode, "..", &PV)

After obtaining PV, the process proceeds to step **410**, which obtains the FID corresponding to the PV. The next step **411** is to add the FID for the PV to the FID chain. The FID for the PV at this point in the example description is FID(local). The FID chain now contains the FIDs for bin and local. The process moves to step **412** where

25 the PV is tested to see if it is the root vnode of the file system name space. If PV is the root vnode, then the FID chain construction is complete and the flow proceeds to step **414**. If PV in step **412** is not the root vnode, then the processing moves to step **413** where the variable tVnode is set to the vnode PV which at this point represents the vnode for the directory local in the presented example path of /usr/local/bin/date. The process now

30 moves to step **409** where the flow of steps to obtain the parent vnode for tVnode, generate a FID, and add the FID to the FID chain repeats. This set of steps (**409-413**)

iterates until the FID chain construction is complete indicated by reaching the root of the file system name space. Once the FID chain is completed as indicated in step **414**, the processing moves to step **415** where the FID chain is inserted into the FID chain cache (FCC) with the first FID in the chain serving as the entry's search key. In the presented example of /usr/local/bin/date, the key is the FID for bin (FID(bin)), which is the variable dirFid as presented in the flow of steps in Figure 4. After inserting the FID chain into the FCC, the process proceeds to step **416** where the flow of steps ends. In step **416** the obtained FID for the target (tgtFid) and the corresponding FID chain for the directory components leading to the target are returned to the invoker.

Figure 5 illustrates the steps involved in locating a FID chain in the FID chain cache. This process uses the FID specification of the terminating component of a directory path to start the search for the FID chain. Step **500** retrieves this FID specification. In step **501**, the FID would be processed into a search hash list to find a FID match. A common practice in computing and software programming for databases is to take the information and process it into a list or index. For example, one means to speed up a search for a specific FID chain would be to divide the FID chains in the cache into lists. After this sorting exercise, some additional processing occurs to determine in which list to start the search.

After selection of the initial list, step **502** begins the search with the first FID chain at the head of the list. Step **503** is a determination of whether the first FID in the chain matches the initial FID retrieved from the FID specification in step **500**. If the first FID does match the initial FID, then step **504** returns a found FID chain and the process ends. If the FID in the list does not match the search FID, step **505** gets the next FID chain in the list. Step **506** determines whether the search has reached the end of the list. If the search has reached the end of the list, then no FID chain is found for the FID in the search. Step **507** returns a no FID found and the process ends. If this list has more entries, the process returns to step **503** and repeats steps **503**, **505**, and **506** as necessary.

Figure 6 shows the flow of steps to flush a cached FID chain from the FCC when an operation has occurred that affects the file system name space. For such an event, information in the cache might be rendered stale. An example of such an event is a rename operation on a directory. A specific example would be to rename /usr/local to

/usr/loc. This would make any cached fid chain containing a FID for local stale. The flow starts in step **600** with the path name (path) describing the file system resource (which would be a path name to a directory file system resource) to be renamed. The flow proceeds to step **601** where a vnode is obtained for the directory. The process moves to
5 step **602** where the FID (dirFid) is retrieved using the vnode. The search through the FCC begins in step **603** starting with the first hash queue in the FCC and proceeds to step **604**, which obtains the first FID chain in the hash queue. In step **605**, all the fid chain's FIDs are compared against dirFid. If any FIDs in the chain match dirFid as tested in step **606**, the FID chain is removed from the FCC in step **607** which returns back to the main
10 flow of step **608**. If in step **606**, there is no match against dirFID, the FID chain is not affected, and therefore remains in the cache. The flow moves to step **608**. In step **608**, the next FID chain in the hash queue is retrieved. If step **608** finds there is a next FID chain in the queue, which is tested in step **609**, the logic returns to step **606** where the next FID chain is processed by steps **606**, **608**, and possibly **607**. If the end of the list has
15 been reached, then step **610** checks to see if there are more hash queues in the FCC to search. If not, the flow moves to step **612** and ends. Otherwise the processing returns to step **604** to search the next queue. At the conclusion of the flow of steps in Figure 6, all FIDs, which could have been rendered stale by the operation on path, are removed from the FCC, as that operation potentially affected the name space of the file system with
20 respect to path.

Figure 7 illustrates a high level architecture of the example Security Classification System (SCS) containing the described techniques of the present invention. In the diagram, box **700** is the repository holding Security Classification Categories (SCCs) which are defined for file system resources using the full path name of the resource. This
25 name is termed a Defined Name, or DN in the example system. The SCDB Manager, shown in box **701** could be in the form of a machine or process on a system which might reside in a network of computers. When the SCS starts on a system where it has been applied, in box **703**, it first extracts the records in the SCDB, obtains a FID for the DN in the record, and creates an entry in the FID to DN mapping database. Each entry has the
30 DN and associated SCC contained in it for the entry's FID which acts as the search key into the database. Box **703** makes use of the techniques described in Figure 1 of the

present invention. Box **705** of Figure 7 represents the programs and users which access file system resources on a system where SCS monitoring runs. File system resources are accessed using Application Programming Interfaces (APIs). Box **704** represents the component of the SCS, which intervenes in accesses to file system resources. It may do this through a variety of techniques, which exist in the practiced art. Examples might include, library replacement, system call table modification, or command replacement. The techniques, in box **704**, utilized the processes in Figures 2, 3, 4, 5 and 6. Box **706** of figure represents a potential location where accounting data collected might be logged.

Figure 8 depicts a pictorial representation of data processing system **800** which may be used in implementation of the present invention. As may be seen, data processing system **801** includes processor **801** that preferably includes a graphics processor, memory device and central processor (not shown). Coupled to processor **801** is video display **802** which may be implemented utilizing either a color or monochromatic monitor, in a manner well known in the art. Also coupled to processor **801** is keyboard **803**. Keyboard **803** preferably comprises a standard computer keyboard, which is coupled to the processor by means of cable **804**. Also coupled to processor **801** is a graphical pointing device, such as mouse **805**. Mouse **805** is coupled to processor **801**, in a manner well known in the art, via cable **806**. As is shown, mouse **805** may include left button **807**, and right button **808**, each of which may be depressed, or "clicked", to provide command and control signals to data processing system **800**. While the disclosed embodiment of the present invention utilizes a mouse, those skilled in the art will appreciate that any graphical pointing device such as a light pen or touch sensitive screen may be utilized to implement the method and apparatus of the present invention. Upon reference to the foregoing, those skilled in the art will appreciate that data processing system **800** may be implemented utilizing a personal computer.

Although the various methods described are conveniently implemented in a general-purpose computer selectively activated or reconfigured by software, one of ordinary skill in the art would also recognize that such methods may be carried out in hardware, in firmware, or in more specialized apparatus constructed to perform the required method steps.

Further, it is important to note that while the present invention has been described in the context of a fully functioning data processing system, those skilled in the art will appreciate that the processes of the present invention are capable of being distributed in the form of instructions in a computer readable medium and a variety of other forms, regardless of the particular type of medium used to carry out the distribution. Examples
5 of computer readable media include media such as EPROM, ROM, tape, paper, floppy disc, hard disk drive, RAM, and CD-ROMs and transmission-type of media, such as digital and analog communications links.

Having thus described the invention, what we claims as new and desire to secure
10 by Letters Patent is set forth in the following claims.

11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201